

### **Remarks**

In the Office Action, the Examiner noted that claims 1-31, 33 and 34 are pending in the application, and that claims 1-31, 33 and 34 are rejected. By this amendment, claims 18 and 23 have been amended. Thus, claims 1-31, 33 and 34 are pending in the application.

Applicant hereby requests further examination and reconsideration of the application, in view of the foregoing amendments.

### **In the Claims**

#### **Objection Because of Informalities**

The Examiner objected to claim 18 because of informalities, and required correction to eliminate concern regarding a negative claim limitation. Applicant has amended claim 18 as requested.

#### **Rejection Under 35 USC 102(b)**

The Examiner rejected claims 1-6 under 35 U.S.C. § 102(b), as being taught by *Popescu et al.*, U.S. Patent No. 5,487,156 (hereinafter *Popescu*). Applicant respectfully traverses.

Broadly speaking, *Popescu* teaches a dynamic register file (DRF) that acts as an instruction queueing mechanism that decouples instruction fetching from instruction execution. Col. 2, lines 25-27. Results of executed instructions are temporarily stored into the DRF before updating the processor state. Col. 2, lines 55-59. *Popescu* also teaches a memory scheduler that controls the DRF and performs resource checking and data flow scheduling out of the DRF. Col. 6, lines 51-54.

With respect to claim 1, the Examiner asserts that *Popescu* teaches a result forwarding cache (RFC), for storing at least one non-store instruction result destined for a user-visible register of the microprocessor and for storing a plurality of store instruction results, referring to *Popescu*'s DRF. Applicant respectfully asserts that *Popescu* does not teach that store instruction results, i.e., store data, are stored in the DRF. Rather, *Popescu* teaches at col. 7, lines 19-20 that the source data of a store instruction to be written to memory (which are referred to as store instruction results or store results or store data in Applicant's specification, see, for example, pg 24, ln 16 to pg 25, ln 15; Fig. 3, elements

314 and 316; Fig. 4, element 416; Fig. 5, note clocks 4 and 5; Fig. 8, note clock 6; pg 35, lines 3-7; pg 40, ln 21 through pg 41, line 7; pg 43, line 21 through pg 44, line 2 (store buffer 188 contains store data, not store addresses that are stored in element 228); pg 44, lines 16-19), unlike the results of non-store instructions, are not stored in *Popescu's* DRF, but rather are stored in *Popescu's* architectural register file (element 17 of Figs. 1 and 2), which is not a result forwarding cache. Rather, *Popescu* teaches at col. 8, lines 60-62 that for store instructions, the store instruction store address, i.e., the memory address to which the store data is to be written, is stored in the DRF RESULT field, not the store data itself. Fig. 2 of *Popescu* shows the architected register file, not the DRF, providing the store data (element 63), which is further taught at col. 8, lines 63-67. That the store data is not stored in the DRF is further confirmed by the fact that, as Fig. 10 shows, the RESULT field is only 39 bits, which is not large enough to store both the store address and the store data. Therefore, *Popescu* does not teach an RFC that stores a plurality of store instruction results and *Popescu* does not anticipate claim 1.

Further with respect to claim 1, the Examiner also asserts that *Popescu* teaches control logic, configured to receive a load/store address match signal and selectively forward one of a plurality of store instruction results from an RFC to a pipeline stage executing a load instruction in response to said match signal, referring to *Popescu's* memory scheduler. Applicant respectfully asserts that *Popescu* does not teach the memory scheduler forwarding a store instruction result from the DRF to a load instruction. First, as discussed above, the DRF does not contain store instruction results and therefore cannot forward store instruction results. Furthermore, *Popescu* teaches at col. 8, lines 28-58 that load instruction data is obtained from memory, not from the DRF. The memory scheduler determines whether a load instruction address matches a store address in the DRF ("unsafe" condition), but for the purpose of delaying obtaining the load data from memory (until a "safe" condition), not for the purpose of forwarding data from the DRF to the load instruction, which the DRF does not do. Therefore, *Popescu* does not teach forwarding store instruction results from an RFC to a load instruction and *Popescu* does not anticipate claim 1.

Applicant respectfully asserts *Popescu* does not anticipate dependent claims 2-6 because they depend from independent claim 1, which is not anticipated by *Popescu* for the reasons discussed above.

### **Rejection Under 35 USC 103**

The Examiner rejected claim 7 under 35 U.S.C. § 103 as being unpatentable over *Popescu* in view of *In re Rose*, 220 F.2d 459 (CCPA 1955). Applicant respectfully traverses the Examiner's rejections. Applicant respectfully asserts *Popescu* in view of *In re Rose* does not obviate dependent claim 7 because it depends from independent claim 1, which is not anticipated by *Popescu* for the reasons discussed above.

The Examiner rejected claims 8 and 9 under 35 U.S.C. § 103 as being unpatentable over *Popescu* in view of *Abramson et al*, U.S. Patent No. 5,606,670 (hereinafter *Abramson*). Applicant respectfully traverses the Examiner's rejections. Applicant respectfully asserts *Popescu* in view of *Abramson* does not obviate dependent claims 8 and 9 because they depend from independent claim 1, which is not anticipated by *Popescu* for the reasons discussed above.

The Examiner rejected claims 10-20 and 26-29 under 35 U.S.C. § 103 as being unpatentable over *Abramson* in view of *Popescu*. Applicant respectfully traverses the Examiner's rejections.

With respect to claim 10, the Examiner asserts that *Popescu* teaches an RFC configured to forward a first plurality of store instruction results. Applicant respectfully asserts that *Popescu* does not teach an RFC configured to forward a first plurality of store instruction results for the reasons stated above with respect to claim 1. Therefore, *Abramson* in view of *Popescu* does not obviate claim 10.

Applicant respectfully asserts *Abramson* in view of *Popescu* does not obviate dependent claims 11-17 because they depend from independent claim 10, which is not obviated by *Abramson* in view of *Popescu* for the reasons discussed above.

With respect to claim 18, the Examiner asserts that *Popescu* teaches a plurality of stages of a pipeline that does not include store buffers subsequent to a first pipeline stage,

wherein the first pipeline stage has a load instruction load address and the plurality of stages have a plurality of store addresses of store instruction data. Applicant respectfully asserts that the Examiner has failed to distinctly point out a stage in *Popescu*'s processor that has a load instruction load address. Applicant respectfully asserts that the Examiner has also failed to distinctly point out a plurality of stages in *Popescu*'s processor subsequent to the stage with a load address that have store addresses and that do not have store buffers. Applicant notes that *Popescu* appears to teach only three pipeline stages in the processor.

Further with respect to claim 18, the Examiner asserts that *Popescu* teaches an instruction shelfer that compares addresses of both store and non-store instructions. The Examiner further asserts that it would have been obvious to a person of ordinary skill at the time the invention was made to incorporate the instruction shelving of *Popescu* in the device of *Abramson*. Applicant can find no teaching, suggestion, or motivation to incorporate *Popescu*'s instruction shelfer into the processor of *Abramson*. The Examiner appears to rely on the fact that the processors taught in the two references are out-of-order processors as a motivation to combine the elements thereof. However, the benefits of comparing load and store addresses to detect storehits is not limited in its usefulness to out-of-order processors, since in-order processors may also benefit from comparing load and store addresses, such as to perform store data forwarding. Therefore, Applicant fails to see how the fact that both *Abramson*'s and *Popescu*'s processors are out-of-order processors provides a motivation to combine the references. Still further, *Popescu* teaches comparing load and store addresses not for the purpose of forwarding store data to a load instruction, as explained with respect to claim 1 above, but for shelving instructions, i.e., for delaying a load instruction execution rather than furthering load instruction execution by forwarding store data to it, as explained above with respect to claim 1. Therefore, the Examiner has not made out a *prima facie* case that *Abramson* in view of *Popescu* obviates claim 18.

Applicant respectfully asserts *Abramson* in view of *Popescu* does not obviate dependent claims 19-20 because they depend from independent claim 18, which is not obviated by *Abramson* in view of *Popescu* for the reasons discussed above.

With respect to claim 26, the Examiner asserts that *Popescu* teaches storing at least one store instruction result and at least one non-store instruction result into a result forwarding cache of a microprocessor. Applicant respectfully asserts that *Popescu* does not teach storing at least one store instruction result and at least one non-store instruction result into a result forwarding cache of a microprocessor for the reasons stated above with respect to claim 1. Therefore, *Abramson* in view of *Popescu* does not obviate claim 26.

Applicant respectfully asserts *Abramson* in view of *Popescu* does not obviate dependent claims 27-29 because they depend from independent claim 26, which is not obviated by *Abramson* in view of *Popescu* for the reasons discussed above.

The Examiner rejected claims 21-25, 30-31, and 33-34 under 35 U.S.C. § 103 as being unpatentable over *Abramson* in view of *Popescu* and in further view of Patterson and Hennessy's Computer Architecture: A Quantitative Approach, Second Edition © 1996 (hereinafter *Hennessy*). Applicant respectfully traverses the Examiner's rejections.

With respect to claim 21, the Examiner asserts that *Popescu* teaches forwarding logic, coupled to receive a virtual match signal for forwarding storehit data in response to said virtual match signal indicating no match between a virtual load address and a plurality of virtual store addresses, prior to generation of a physical match signal. Applicant respectfully asserts *Popescu* does not teach forwarding logic for forwarding storehit data, as described with respect to claim 1. Furthermore, Applicant can find no distinction in *Popescu* regarding virtual and physical addresses of load or store data, nor regarding comparison of same. Furthermore, as discussed with respect to claim 18, Applicant can find not teaching, suggestion, or motivation to combine the references, including the fact that *Popescu* and *Abramson* both teach out-of-order processors. For these reasons, *Abramson* in view of *Popescu* and in further view of *Hennessy* does not obviate claim 21.

Applicant respectfully asserts *Abramson* in view of *Popescu* and in further view of *Hennessy* does not obviate dependent claim 22 because it depends from independent claim 21, which is not obviated by *Abramson* in view of *Popescu* and in further view of *Hennessy* for the reasons discussed above.

With respect to claim 23, the Examiner asserts that a person of ordinary skill in the art at the time the invention was made would have recognized that incorporating the stall of *Hennessy* would preserve the correct execution pattern when accessing slower memory. Applicant respectfully disagrees. The assertion of the stall signal recited in claim 23 is not related to accessing slow memory, but rather to providing correct data after erroneously speculatively forwarding storehit data. Claim 23 recites a situation in which store instruction data whose address matches a subsequent load address (the storehit data) is still present in the microprocessor (i.e., the processor has not yet written the store data to its destination device, such as a memory or I/O device) and the load address is in a non-cacheable region. Since the load instruction depends on the non-cacheable storehit data being written to memory before it can be loaded back into the processor, according to *Hennessy* a data hazard is present. *Hennessy* teaches stalling a microprocessor pipeline in response to a data hazard – *which presumes the data hazard has already been detected* – until the hazard has gone away, which in this case would be until the processor writes out the storehit data. However, the invention of claim 23 does not even detect the hazard until it has already speculatively forwarded the storehit data to the load instruction from within the processor – without waiting to determine whether a hazard exists, and upon detection of the hazard, the invention subsequently stalls the pipeline for the purpose of correcting the violation of the now-detected hazard. Therefore, Applicant fails to see the motivation suggested by the Examiner to incorporate the stall of *Hennessy* into the device of *Abramson*. For these reasons, Applicant respectfully asserts that *Abramson* in view of *Hennessy* does not obviate claim 23.

Applicant respectfully asserts *Abramson* in view of *Hennessy* does not obviate dependent claims 24-25 because they depend from independent claim 23, which is not obviated by *Abramson* in view of *Hennessy* for the reasons discussed above.

With respect to claim 30, Applicant respectfully asserts, for the reasons discussed above with respect to claim 21, that *Abramson* does not teach detecting a virtual aliasing condition with respect to a load address and one of a plurality of store addresses based on a physical address comparison between the load address and the plurality of store addresses after speculatively forwarding storehit data from a first microprocessor stage to

a second microprocessor stage based on a virtual address comparison. Therefore, *Abramson* in view of *Popescu* and in further view of *Hennessy* does not obviate claim 30.

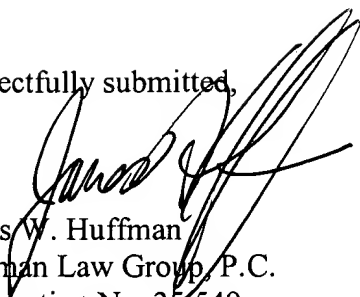
Applicant respectfully asserts *Abramson* in view of *Popescu* and in further view of *Hennessy* does not obviate dependent claims 31 and 33 because they depend from independent claim 30, which is not obviated by *Abramson* in view of *Popescu* and in further view of *Hennessy* for the reasons discussed above.

With respect to claim 34, the Examiner asserts that a person of ordinary skill in the art at the time the invention was made would have recognized that incorporating the stall of *Hennessy* would preserve the correct execution pattern when accessing slower memory. Applicant respectfully disagrees for the reasons stated above with respect to claim 23. Therefore, *Abramson* in view of *Popescu* and in further view of *Hennessy* does not obviate claim 34.

For all of the reasons advanced above, Applicant respectfully submits that claims 1-31, and 33-34 are in condition for allowance. Reconsideration of the rejections is requested, and Allowance of the claims is solicited.

Applicant earnestly requests the Examiner to telephone him at the direct dial number printed below if the Examiner has any questions or suggestions concerning the application or allowance of any claims thereof.

Respectfully submitted,



James W. Huffman  
Huffman Law Group, P.C.  
Registration No. 35,549  
Customer No. 23669  
1832 N. Cascade Ave.  
Colorado Springs, CO 80907  
719.475.7103  
719.623.0141 fax  
jim@huffmanlaw.net

Date:

8-27-04

"EXPRESS MAIL" mailing label number E0002522526US. Date of Deposit  
8-27-04. I hereby certify that this paper is being deposited with the U.S.  
Postal Service Express Mail Post Office to Addressee Service under 37 C.F.R. §1.10 on  
the date shown above and is addressed to the U.S. Commissioner of Patents and  
Trademarks, Washington, D.C. 20231.

By:

